SB^{3}C2023
Summer Biomechanics, Bioengineering and Biotransport Conference
June 4-8, Vail, CO, USA

TESTING USABILITY OF 3D VISUALIZATION SHARING INTERFACES FOR CFD

Noah Egnatis (1), Zainab Husain (2), David A. Steinman (1,2,3)

(1) Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada
(2) Department, University, City, State, Country
(3) Department, University, City, State, Country

INTRODUCTION

Easy access to explore-able blood flow data (data that can be quickly visualized and adjusted to the users liking) is crucial for the task of bridging technical gaps between communities and involving researchers from different disciplines in CFD research. This study aims to identify practical methods for sharing adjustable 3D visualizations between researchers. An aortic helicity visualization was created from 4DMRA data and was shared using 3 techniques: a Google Colab notebook, a flask webpage, and a Docker contained notebook [add reference to gitlab]. The working Google Colab Notebook can be accessed here:[add github colab link]. Researchers with varying computational skill levels were surveyed after accessing data through each technique. Each technique was evaluated on ease of setup and interactivity for the viewer, and ease of deployment for the visualization developer.

BACKGROUND

Data Accessibility: While robust visualization tools such as Paraview can be used for interactively viewing 3D visualizations, setup requires multiple steps, including large data file sizes and software downloads. The data interaction often requires familiarity with the software's interface. As a result, it is commonplace to share 3D visualizations as static 2D images due to ease of access (e.g. sharing 2D screenshots through email). Certain academic journals such as the Journal of Fluid Mechanics have provided recent solutions to this problem through cloud hosted jupyter notebooks [add citation]. These solutions involve their own trade offs such as the fees associated with cloud-based hosting services and the resulting data size limits. Thus, different situations and user demographics may be well suited for different data sharing strategies which is the main motivation for this study.

Sample Dataset: The goal of the visualization sample case was to explore the helical flow behaviour in a human aorta. Existence of helical flow might facilitate blood flow transport and reduce risk of atherosclerosis, thus there is clinical interest in observing high helical flow (healthy) vs low helical flow (unhealthy) patients [cite a Torino paper]. When viewing multiple flow properties simultaneously, such as isosurfaces and streamlines, it is beneficial for the viewer to be able to alter the visualization independently such as adjusting the camera angle and altering the plotting thresholds. Designing visualizations to allow for interaction increases the viewers agency and more closely mirrors the medical data viewing experiences of clinically oriented research collaborators such as medical imaging specialists [cite lucas's dicom paper?].

METHODS

The 4DMRA data was preprocessed in Paraview where velocity streamlines and Local Normalised Helicity values were computed and saved as vtu files. All techniques utilized the PyVista python library as its visualization basis.

Technique 1: Google Colab: Google Colab is a type of Jupyter notebook hosted on Google's cloud servers. For setting up shareable visualizations on Colab, the developer must store the data in an online storage service, in our example GitHub, and simply specify the data location within the notebook. This method is very quick and free to deploy but is mainly suitable for lightweight data-sets. The main drawback to Colab is the limitations on 3D rendered objects. While 3D objects (rotate-able through mouse dragging) exist in some libraries available in Colab, 3D support is currently unstable and, in particular, not recommended for heavy meshes.

From a user perspective, a visualization created on colab can be accessed through a web-link and required no software setup on the users local computer. The visualization is displayed as a static 2D image that can be updated by the user through editing the parameter variables, such as viewing angle directly in the code. One question of interest for this study was to compare the preference of mouse rotate-able 3D objects in comparison to a 2D image that can be re-plotted by the user, a low tech solution which still allows some interactivity.

Technique 2: Docker (Containerized Notebook): Downloading and running a Jupyter notebook on a local computer can result in hardware related graphics errors for 3D rendered objects. For this reason, it is best practice to run local notebooks within a container to standardize visualization rendering for different computers [add citation about containerized notebooks]. To set up the Docker visualization the developer is required to have knowledge of container based development such as creating Docker images (containing the notebook and the data) and pushing the images to Docker Hub.

For the user, this technique requires the most setup time and technical skill, requiring the installation of Docker software and using terminal commands to start the container before the notebook can be viewed. In our survey example, adjusting thresholds involved editing the notebook code as before. However, this method had the added benefit of 3D object rotation as the customizability of docker allowed us to specify the necessary 3D Graphics back-end libraries. This technique is well suited for situations involving large data-sets as all computations are done locally. The caveat is that this places requirements on the user to have a suitable personal computer. There was no cost associated with this method.

Technique 3: Flask Web-page: Utilizing a web-page to display visualizations, such as the Flask example, requires all of the knowledge of the previous two techniques plus knowledge required to develop a web application and deploy the application through a chosen cloud server (e.g. Heroku, AWS). The user was able to access the visualization through a web-link, use HTML widgets for changing thresholds and 3D rotate the object.

Table 1: Summary table of visualizating techniques.

	Colab	Docker	Flask
Ease of deployment	Easy	Intermediate	Advanced
Rendering location	Server	Local	Server
Interface	Notebook	Notebook	HTML Webpage
User setup	Web link	Software download	Web link
Threshold adjusting	Editable code	Editable code	HTML widgets
Object rotation	replotable 2D	Mouse rotatable 3D	Mouse rotatable 3D
Cost for sample dataset	Free	Free	Priced

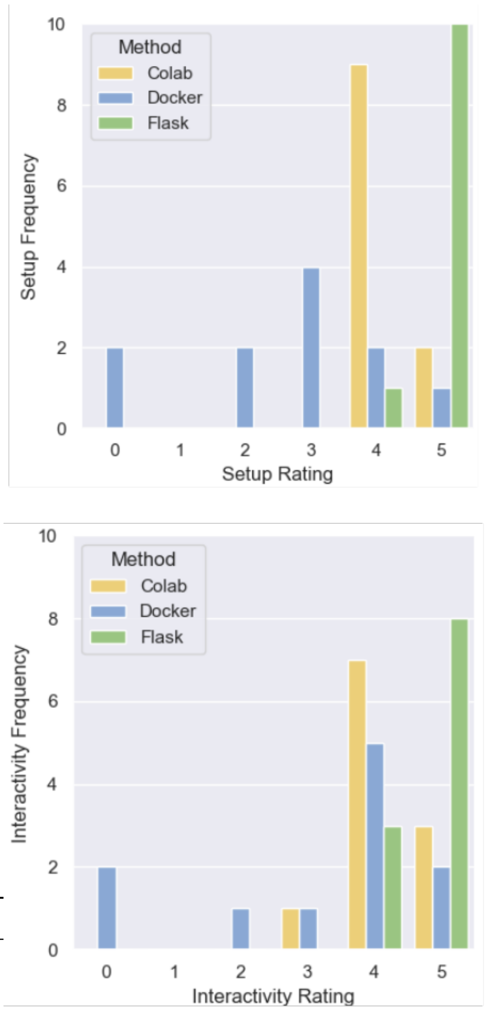

Figure 1: Figure caption centered below the graphic.

Survey Design

Study participants ranged in educational level including Masters and PhD students, Post Docs and Professor's. Prior to the visualization exploration activity, each participant was surveyed on their prior experience using the data exploration tools. Participants described their familiarly is 4 skill areas: python, notebooks, terminal commands and Docker, by categorizing themselves as either novice (never used this tool), beginner (occasionally used this tool) and experienced (comfortable using this tool) for each of the skills.

RESULTS

Participants were generally experienced at using python and beginners at using notebooks and the terminal commands. Only 1 participant had prior experience using Docker. During the survey all 11 participants were able to access visualizations shared through Flask and colab. 2 participates were not able to access visualizations through Docker on their personal laptops due to technical challenges during installation. Flask had the higher overall ratings for both ease of setup and interactivity. Docker had the lowest ratings for both categories.

DISCUSSION

Google Colab could be a quick solution for rapid sharing between colleagues, but would be much preferred with stable 3D rendering support. 3D rendering and widget support for more customizable cloud hosted notebook services, such as CoCalc, should be explored for future iterations, although cost should be considered. Initial setup of docker containers was inconvenient for computer beginners due to difficulty installing and configuring on non-unix based systems [], but could be a viable option for motivated users for continued use.

Overall, users had a strong preference for quick setup time and rotatable 3D plots as demonstrated in the Flask Web-page example. The visual simplicity of the web-page displaying only the data plot and control widgets was also noted as a benefit by participants. The would be the optimal solution for users with minimal technical background and is a use case where it is worth investing the extra time, money and effort for the developer.

While the demographic of this study was mainly CFD researchers due to time constraints, the accessibility of the visualization interface is particularly crucial for involving clinical researchers in CFD research. Clinical user testing is a future goal for evaluating the Flask web-page method.

ACKNOWLEDGEMENTS

Acknowledge grant or other funding support or the assistance of others as appropriate.

REFERENCES

References should be arranged in numerical order according to the citation sequence in the text. Each reference should include the last name of at least the first author followed by his/her initials, the journal name, volume, pages and year. You may include more detailed reference information if space allows.